General Temporal Knowledge for Planning and Data Mining

Robert Morris (1) Lina Khatib (2)
(1) Research Institute for Advanced Computer Science
(2) Kestrel Technology
NASA Ames Research Center
Moffett Field, CA 94035
{morris,lina}@ptolemy.arc.nasa.gov

Abstract

We consider the architecture of systems that combine temporal planning and plan execution and introduce a layer of temporal reasoning that potentially improves both the communication between humans and such systems, and the performance of the temporal planner itself. In particular, this additional layer simultaneously supports more flexibility in specifying and maintaining temporal constraints on plans within an uncertain and changing execution environment, and the ability to understand and trace the progress of plan execution. It is shown how a representation based on single set of abstractions of temporal information can be used to characterize the reasoning underlying plan generation and execution interpretation. The complexity of such reasoning is discussed.

1 Introduction

As AI systems continue to mature, they are more often found supporting real world applications. These applications commonly require the performance of a multitude of intelligent tasks. For example, planning systems [1] are currently often found in automated or limited mixed-initiative systems that combine plan generation and execution [18], [5]. Designing and developing representations for such *continual planning systems* raise challenging issues in temporal reasoning. For example, can a single representation of time be used for both plan generation and automated control of execution? Similarly, can a single representation of time be used by an automated system to formulate long-range plans to be

[1] Although plans are often distinguished from schedules in the literature, the distinction is not important for our purposes here; consequently, the notion of "planning" used here should be interpreted broadly enough to include scheduling.

executed automatically, and to interpret and summarize their execution based on stored traces? It is the latter question that is addressed and answered here.

The planning problems of interest here can be formulated as those involving a number of tasks, many to be executed a number of times during the planning period, as well as constraints on those tasks, and possibly an objective to be optimized. There is uncertainty and uncontrollability in the execution environment, as well as incompleteness in the temporal domain model. The planner must continuously generate plans for executing these tasks continuously over time.

The following are examples of this kind of planning problem:

1. **Telescope Observation Scheduling** [3] Telescope time for the purpose of observing time-varying phenomena (e.g. eclipsing binary stars) is requested by an astronomer. An astronomer's scientific agenda (e.g., to fill out a light curve for a binary star system), imposes various constraints; for example, on the number of observations and on the number of nights between successive observations. Thus, an astronomer might request that a given number of repeated observations (specified by an ideal and minimum occurrence count) be executed within a given time window with a given time gap between observations. The ideal gap (in days) is specified either with a fixed gap length or a gap probability distribution (in order to reduce aliasing in the data or determine the period of a recently discovered variable star). An example of a gap probability distribution would be expressed as "gaps should be randomly selected with a uniform probability from the set { 0 days, 1 day, 2 days }".

2. **Maintenance Scheduling of Power Generating Units** [9] A power plant consists of a number of power generating units which can be individually scheduled for preventive maintenance. The duration of each unit's maintenance period and the power demand of the plant are known. The maintenance scheduling problem is to determine the duration and sequence of outages of power generating units over a given period, subject to various constraints.

3. **Planning Autonomous Spacecraft Operations** In this planning problem, [18], there are a set of tasks involving some operation of the spacecraft, each associated with an interval of time. Each interval of a given type must satisfy a contextual constraint (called a *compatibility*) that is specific to the type. The contextual constraint surrounds the given primary interval with a set of satellite intervals of specified types that stand in specified temporal relationships to the primaries. For example, every occurrence of a *Thrust(B)* interval (thrust in direction B) must be *contained_by* some occurrence of a *Point(B)* (point in direction B) interval.

The formulation of each of these problems involves a specification of a set of events including many that will, in any solution to the problem, have multi-

ple occurrences (e.g., observing a particular star, performance of a maintenance task for a generating unit, thrusting in a certain direction). The total number of occurrences of such an event cannot always be said to be known in advance; for example, it might be the case that the system is to maximize the number of occurrences of the event. Similarly, constraints on those events are meant to apply to any occurrence of those events; for example, in the telescope problem, the gap constraint is meant to apply to *each* gap between observation occurrences. In this paper, constraints associated with events that will have multiple instances in any solution are called *general constraints*, since they are naturally formalized as quantified formulae over temporal objects.

This paper addresses potential requirements of temporal reasoning systems that perform continual planning. Specifically, to ensure the ability to continually generate useful and robust plans, it might be necessary to have a mechanism for evaluating segments of plans previously generated, and, where necessary, update the temporal domain model used in planning. Relatedly, there should be a mechanism for automatically summarizing and interpreting the executions of plans, for purposes such as

- *verifying* a temporal model of the domain, to ensure that the constraints identified for the problem are correct and complete; and
- *identifying interesting episodes* that reveal unexpected features of the environment, which can be applied to refine the domain model.

A simplified architecture for a system for continual planning is found in Figure 1. This figure shows a set of requests that certain tasks be performed, at certain times, which are collected together in a *request database*, which is, in turn, compiled into a set of tasks to be performed. The requests might be inputs from human users, or some other autonomous system. A *plan generator* consults this database, as well as a *temporal domain model*, and produces input in the form of a *task database*. The temporal planner generates a complete plan (represented in the figure by a graph of temporal dependencies), or determines that its input is inconsistent.

The next phase of the planning process involves the execution of a plan. A *plan runner* continuously consults the current plan and executes the tasks that are currently active, i.e., whose conditions for execution are met. (The actual plan execution process might be quite complex, involving a number of other human or machine agents.) A trace of the plan execution is kept in a *execution log*, time-stamped information about the tasks actually performed. The log database might also contain information about the effects of the plan, which are observed in the world. An *execution analysis* tool consults this log, and generates summaries of the plan execution, and also potentially generates updates to the temporal domain model. Note that since the focus here is on time, there are simplifications to the overall picture applied here. Planning in general involves the manipulation of non-temporal constraints, such as resource

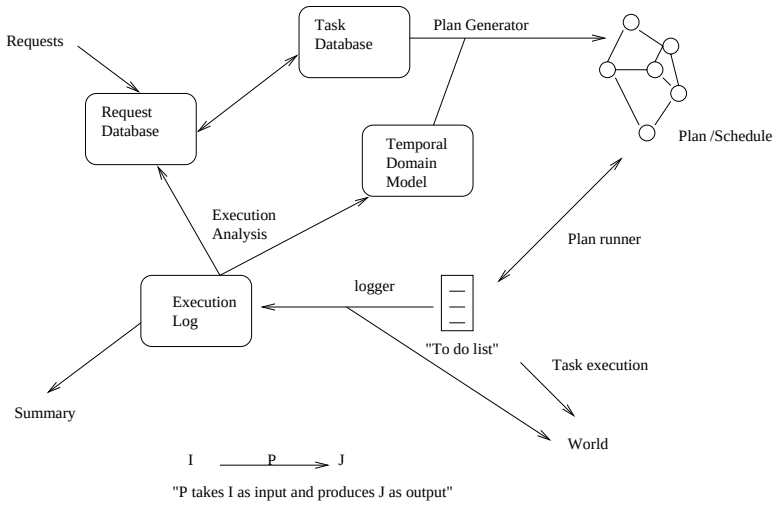

Figure 1: Intelligent continuous planning and execution

constraints, as well as temporal ones. We abstract from considerations of non-temporal information, but maintain that they can be added by generalizing the proposed framework to include them.

This paper proposes a single representation of time to support the capabilities just described. The framework integrates previous work in temporal reasoning about repeating events, [17], [14], [15], as well as in temporal constraint reasoning. It also proposes the use of temporal data mining technology in order to interpret the temporal information found in plan executions. We demonstrate how a single formalism can simultaneously support plan generation from general temporal constraints and execution interpretation from execution logs. The basis for this formalism is a straight-forward and intuitive collection of abstractions of temporal information, based on terminology that is common in the research literature on constraint-based temporal reasoning. The pivotal level of abstraction, based on the concept of a "profile", is a concise representation of distance or temporal order information among sets of intervals. Profiles exhibit patterns that can be used for determining consistency of a set of constraints, forming the basis for solving the planning problem, or for detecting useful temporal patterns, useful in formulating expressions of general temporal knowledge from raw, time stamped data.

The remainder of this paper is structured as follows. In section two, there is a discussion of the general nature of temporal reasoning, for the purpose of setting the stage for the remainder of the paper. There is also, in the same section, a brief summary of the components of the representational mechanism for

reasoning about general temporal knowledge. In section three, there is a concise but detailed discussion of the use of this framework for turning general temporal knowledge into input to temporal constraint reasoners. Finally, in section four, there is a discussion of how a data mining tool can be designed to generate general temporal knowledge from traces of executions stored in a flat format called *instantiations*, using profiles as a concise intermediate representation.

2 Framework

Pure temporal reasoning is reasoning about temporal entities such as points or intervals. The reasoning is used primarily to infer knowledge about temporal *locations*, i.e., when something happens, *durations* (i.e., how long something lasts), or *temporal orderings* (i.e., what follows what). Temporal reasoning should be distinguished from a more broad class of reasoning, which we will call *reasoning about temporal entities*. By a temporal entity is meant, roughly, any entity that can be viewed as being extended in time. Events are the simplest example of a temporal entity, but there may be others, such as the state of an object, or the truth of a proposition.

It may seem obvious, but it is worth stating explicitly: one can reason *about* temporal entities without the reasoning being temporal in nature. The Yale Shooting Problem, for example, involves reasoning about events, but the reasoning is not temporal, since the orderings and durations are either known (hence, no need to infer them) or unimportant; rather the reasoning is about the causal effects of firing a gun. Similarly, much of the body of research that goes under the title *temporal data mining* does not involve mining temporal information. For example, time series analysis is a form of temporal data mining, but the knowledge gained from this analysis is not temporal in nature, i.e., it is not about durations or temporal orderings. That part of temporal data mining which is purely temporal sometimes goes under the heading *mining for interesting episodes* [10].

2.1 Pure temporal reasoning

The focus here is on pure temporal reasoning for planning, i.e., reasoning about things like durations and temporal orderings for the purpose of generating tasks to perform. Of course, on some level, it is artificial to distill pure temporal reasoning from any real reasoning problem involving temporal entities. In practice, a model of time is inextricably linked to other kinds of world knowledge, such as knowledge about space or causality. Nonetheless, traditionally it has been found useful in mathematics and computer science to study properties of time independently. From a computational standpoint, the primary purpose of isolating temporal knowledge is to study the complexity of temporal reasoning in order to isolate tractable instances of temporal reasoning problems, or to

develop efficient approximate algorithms for solving such problems. Temporal reasoning is relevant for proving properties of concurrent systems [24], querying temporal information [11], or for solving hard planning problems.

Temporal reasoning requires an underlying theory, or model, of time. A theory of time is a set of assumptions, expressed using a temporal logic, or as a restricted first-order logic. A theory expresses unassailable truths about the domain of discourse, addressing things like whether time is point- or interval-based, discrete or continuous, infinite or finite in either the past or future, and linear or branching. Such assumptions may be explicitly stated, or embedded in the rules and procedures for manipulating propositions about time. A theory might also state whether the set of occurrences of a temporal entity can be finite or infinite, as well as the granularities of time (minutes, days, weeks, etc.) that can be expressed in the language. Finally, as we will observe later, a model of time may formulate answers to common sense questions like "what does it mean, in general, for two events to occur close together, or frequently"?

The formal approach to be taken here, reflecting the focus on time used in planning, can be viewed as an example of a first-order treatment of time. In what follows, propositions about time quantify over intervals. Time is linear and discrete, and intervals are sets of integers. Events are the only kind of temporal entity in this model; they can be classified into types. Each event can occur arbitrarily often; an occurrence of an event is described by the interval associated with it. Quantification is sorted based on the type of temporal entity. For example, in the expression $\forall I_n \in I_E$ I_E denotes all the occurrences of an event of type E. Intervals, equivalently event occurrences, have distinguished start and end times, which can be denoted by the functional expressions $s(I_n), e(I_n)$. For simplicity, intervals in a set I_E are assumed to be totally ordered by the temporal relation *before* ($<$); the expression I_j (the jth interval, equivalently, the jth occurrence of some event) is used to recover this ordering. Where there is no ambiguity, we often abbreviate $I_n \in I_E$ to $I_n \in I$.

General temporal knowledge provides information about:

- the *cardinality* of I_E, i.e., the number of times E occurs;
- temporal orderings (before, after, overlap, containment, etc.) between occurrences; and
- temporal distances between occurrences.

Quantification over intervals is an obvious mechanism for formulating general temporal knowledge. For example, in addition to being able to state that a pair of occurrences I_n and J_m are ordered by the relation *meets*, we wish also to be able to say that for each occurrence $I_i \in I_E$ there is an occurrence $J_j \in J_F$ such that I_i and J_j are ordered by the relation *meets* (in English, this would be expressed as **Es only meet Fs**). Similarly, in addition to saying that the distance between $s(I_1)$ and $s(J_1)$ is 3 time units, the distance between $s(I_2)$ and

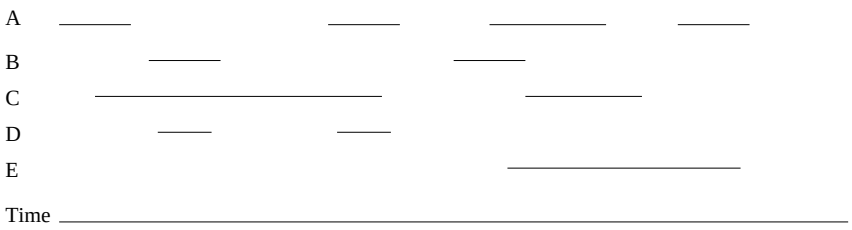

Figure 2: A set of instantiations.

$s(J_2)$ is 4 time units, and $s(I_3)$ and $s(J_3)$ is 2 time units, we also want to say that For each $I_i \in I$ there is a $J_j \in J$ such that the distance between $s(I_i)$ and $s(J_j)$ is in the interval $[2, 4]$.

2.2 Representation

The framework here is based on the following succession of abstractions of temporal information:

- An *instantiation* of a repeating event;
- A *profile* of a repeating event;
- A *profile summary*; and
- A *specification*.

An example of an instantiation is found in Figure 2. There are five events, labeled A-E, each with a finite number of non-overlapping occurrences, with possibly different durations. Notice that the information contained in the figure is equivalent to one in which the temporal information is stored in a table or relation, where each tuple in the relation has the form (X, n_1, n_2), where X is the event type, and $[n_1, n_2]$ is the associated interval describing the time of the occurrence.

Given an instantiation of a repeating event, duration (temporal distance) information can be completely represented in the form of a set of *profiles*, of which five can be distinguished: four in terms of time point combinations (end-start, start-start, start-end, and end-end), and one which displays ordering information about the intervals. For a finitely repeating event, each profile can be viewed as a matrix. Figure 3 shows an instantiation of a repeating event with three occurrences, and three profiles associated with it. Each value in a profile is the difference $x(I_j) - y(I_i)$, where $x, y \in \{s, e\}$, and I_m is the m^{th} occurrence of I. By convention these profiles are referred to using an expression of the

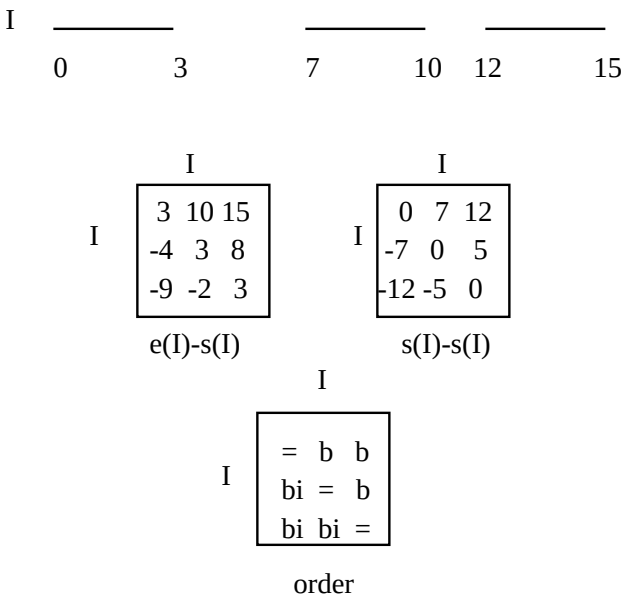

Figure 3: An instantiation of a repeating event and three profiles.

form $x(J) - y(I)$, where the I's index the rows of the profiles, and the J's the columns. The other profile in the figure, called the *order* profile, summarizes all the qualitative temporal relationships between pairs of occurrences of I.

The notion of profile can also be used to represent distance or order information about pairs of occurrences of distinct repeating events I and J. Each value of a distance profile is a difference $x(I_j) - y(J_k), x, y \in \{s, e\}$ between an terminal point (start s or end e) of an occurrence of I and one of J. A value $P_{I,J}[i, j]$ of an order profile is the Allen relation between I_i and J_j. We refer to each of these profiles as part of the *relative profile* of two repeating events. Again, five profiles can be distinguished, and, assuming both repeating events have finite cardinality, the information can be depicted in the form of a matrix. Figure 4 illustrates relative profiles.

Profiles have patterns that emerge from the underlying temporal structure of the repeating event. Such patterns provide the basis for inferring new temporal knowledge, or for extracting temporal information in response to a query. For example, in a distance profile of a finite repeating event with no overlapping occurrences, each row is a sequence of either monotonically increasing or decreasing values, as is each column. We call this the *monotonicity requirement* of profiles. Note also that every qualitative profile for a non-overlapping sequence of intervals consists of a lower-left triangle of b relations, an upper-right triangle of bi relations, and a diagonal of $=$. We call this the *tri-regional* require-

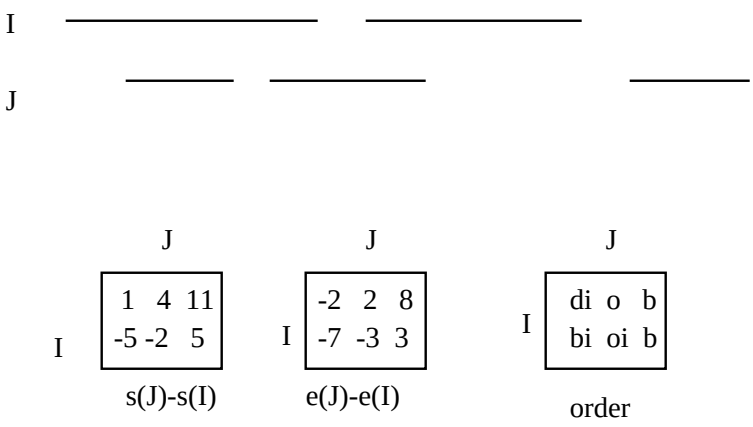

Figure 4: An instantiation of a pair of repeating event and three relative profiles.

ment for admissible qualitative profiles. Relative profiles of pairs of finite, non-overlapping repeating events have monotonicity and tri-regional requirements for admissibility as well. In addition, it is possible to characterize admissibility for *sets* of profiles in terms of adherence to constraints imposed by two *profile operations*, inverse and composition. The *inverse* $P_{I,J}^{-1}$ of a distance profile $P_{I,J}$ where $P_{I,J}[i,j] = x(J_j) - y(I_i)$ is the profile $P_{J,I}$ where $P_{J,I}[j,i] = y(I_i) - x(J_j)$ (I and J not necessarily distinct). In matrix format, the inverse of a profile is the negative transposed matrix, i.e., the one that results when the rows and columns are reversed, and the corresponding distance values negated.

Profile composition $\circ$ is defined between pairs of profiles $P_{I,J}, P_{J,K}$, where $P_{I,J}[i,j] = y(J_j) - x(I_i)$ and $P_{J,K}[l,m] = x(K_m) - z(J_l)$, $x, y, z \in \{s, e\}$. The result of $P_{I,J} \circ P_{J,K}$ is a profile $P_{I,K}$. A set P of profiles is admissible with respect to composition if, for any subset of P consisting of three profiles of the form $P_{I,J}, P_{J,K}, P_{I,K}$ $P_{I,K}[i,k] = P_{I,J}[i,j] + P_{J,K}[j,k]$, for each $j = 1 \ldots \|J\|$, where $\|J\|$ is the number of subintervals of J. An arbitrary set of profiles for a collection of non-overlapping repeating events is *admissible* if each distance profile in the set adheres to the monotonicity requirement for profiles, each qualitative profile is tri-regional, and the set is admissible with respect to inverse and composition.

Given an admissible profile, it may be useful to summarize information contained in it. A *profile summary* is a description about a subset of the values contained in the profile. For example, a sentence like *It took all of the three group meetings fifty days to complete*, says something about the specific distance $e(gm_3) - s(gm_1)$, a single value in a profile. Alternatively, saying something like *Each J finished 5 hours after the completion of some I* says something about a set of distances of the form $e(J_j) - e(I_i)$. Finally, to say *none of the I's and J's overlap* is to say something about every value in a qualitative profile associated with I and J.

We say that a profile *satisfies* a summary. For example, the left-most profile in Figure 4 satisfies the summary *Every J starts less than six time units after some I*. In general, P satisfies s by virtue of a set of values in P. If $P_{I,J}[i,j]$ is such a value, then I_i and J_j will be said to be *correlated* with respect to s. Finally, a set S of profile summaries for a set $\mathcal{E}$ of repeating events will be called a *specification* of $\mathcal{E}$.

This completes the description of the representational framework that will be used for both general constraint processing and execution summarization. The following sections formalize each reasoning task within this framework.

3 Temporal Reasoning with General Constraints for Planning

To solve the temporal planning problem, a specification (set of summaries) of profiles will serve as input to a Repeating Event CSPs (RE-CSPs) [17]. An RE-CSP is a CSP in which the variables stand for features of profiles, and summaries collected into a specification are viewed as constraints. Solving an RE-CSP consists of generating a set of admissible profiles that satisfy all the summaries in the specification.

Here is an example of an RE-CSP. Given a set $\mathcal{I}$ of events, the set of variables in the RE-CSP is defined as $\{N(I), E(I), D(I), \forall\ I\ \exists\ J\ D^{I,J}_{x,y}\}, I, J \in \mathcal{I}, x, y \in \{s, e\}$. The variable $N(I)$ is used to constrain the number of occurrences of a repeating event I. Thus the constraint $N(I) \in [3,6] \cup [10,20]$ states that the number of occurrences of I is between either 3 and 6, or between 10 and 20. $D(I)$ denotes the duration of an arbitrary occurrence of I; thus $D(I) \in [4,6]$ abbreviates $\forall\ I_i \in I\ e(I_i) - s(I_i) \in [4,6]$. Informally, this says that each occurrence of I takes between 4 and 6 time units to complete. Third, let $E(I)$ stand for the distance between the end of the last occurrence of I and the start of the first (called the *extent* of a repeating event). The constraint $E(I) \in [30,50]$ thus states that all of I should complete within 30 and 50 time units. Fourth, let $\{\forall\ I\ \exists\ JD^{I,J}_{x,y}, x, y \in \{s, e\}\}$ be a set of variables that stands for the distance between the start or end of every I and the start or end of some J. Thus, the constraint $\forall\ I\ \exists\ JD^{I,J}_{s,s} \in [4,10]$ abbreviates the first-order formula $\forall\ I_i \in I\ \exists\ J_j \in J\ s(J_j) - s(I_i) \in [4,10]$, and says informally that every I should start between 4 and 10 time units before the start of some J. Finally, a variable of the form $\forall I\ \forall J\ R(I,J)$, where R is an Allen relation, refers to all the values of a qualitative matrix. The constraint $\forall\ I\ \forall\ J\{b, bi\}(I,J)$ abbreviates the expression $\forall\ I_i \in I\ \forall\ J_j \in J\ (I_i\ b\ J_j)\ \vee (I_i\ bi\ J_j)$, and states that there is no overlap between any I and any J.

3.1 Solving via concretization into a TCSP

A temporal planning problem can clearly be viewed as the problem of transforming specifications into consistent instantiations. A *consistent* instantiation is one all of whose profiles satisfy each of the summaries found in a specification. We divide the operations involved in transforming specifications into temporal plans into the following two-step operation:

1. "Concretization" a specification, and
2. Solving the resulting CSP.

The notion of *concretization* was first introduced in [17]. Intuitively, it is the result of transforming an RE-CSP specification by assigning numbers to all number variables and establishing correlations between pairs of sub-intervals involved in a binary RE-CSP relation. Formally, for binary relations between I and J of the form $\forall I\ \exists J\mathcal{R}$, a correlation is a total mapping of indices of subintervals from I into indices of subintervals of J. Thus, correlations assume that the cardinality of I and J have been established. For example,

$$S = \{N(I) \in [1,5]; N(J) \in [3,6]; D(I) \in [1,2]; \forall\ I\ \exists J D^{I,J}_{s,s} \in [2,4]\}$$

is a simple RE-CSP specification. Given the assignment $N(I) = 4; N(J) = 5$, and the relation in S between I and J, a correlation $cor_{I\to J}$ is a set of pairs of indices into sub-intervals of I and J. One such correlation can be written

$$cor_{I\to J}(1) = 1; cor_{I\to J}(2) = 1; cor_{I\to J}(3) = 2; cor_{I\to J}(4) = 5.$$

The specified binary relation is transformed, given the number and correlation assignment, into the conjunction

$$s(J_1) - s(I_1) \in [2,4] \wedge s(J_1) - s(I_2) \in [2,4] \wedge$$

$$s(J_2) - s(I_3) \in [2,4] \wedge s(J_5) - s(I_4) \in [2,4].$$

A *concretization* of a RE-CSP specification S is a description of the result of this transformation, for all number and relational constraints in S. For example, the concretization (in predicate calculus notation) for the current example is

$$N(I) = 4 \wedge N(J) = 5 \wedge s(I_1) - s(J_1) \in [2,4] \wedge$$

$$s(I_2) - s(J_1) \in [2,4] \wedge \ldots \wedge s(I_4) - s(J_5) \in [2,4].$$

Viewing a concretization C as a conjunctive formula, C is *consistent* if there is an assignment to each variable $x(I_k)$ that appears in C that makes C true.

By a *trigger* is meant the set of number assignments and correlations that produced a given concretization. We write $S_T = C$ to describe the result of applying a trigger T to a specification S. Since each RE-CSP specification induces

a set of triggers, there is a one-to-many relationship between specifications and concretizations. Since not all resulting concretizations are consistent, the search problem arises of finding a trigger (or all triggers) that produces a consistent concretization. Furthermore, as shown in [14], the size of the search space of triggers is potentially very large, dominated by the number of possible ways of mapping finite sets into finite sets as implied by the $\forall\exists$ logical form of the binary constraints. The skeletal form of an algorithm for determining the consistency of an RE-CSP is the following, where $\mathcal{T}$ is the set of all triggers of S.

input : a specification S of an RE-CSP;
output : a consistent concretization C if one exists.
begin
 for each $T \in \mathcal{T}$
  $C := S_T$;
  if *consistent*(C) **then return** C;
 return fail
end

Once a consistent concretization has been found, the resulting problem can be solved by standard methods. For example, a concretization of certain types of RE-CSP can be viewed as a Simple Temporal Problem (STP) [4], as demonstrated in [15]. The concretization into a STN, for the specification S found in the previous section, and using the example trigger discussed there, is found in Figure 5. In the figure, there are four pairs of nodes representing start and end points of I, and five for J. There are also labeled arcs between the end points and the start points of J, concretizing the constraint on duration found in the specification. Finally, there are labeled arcs between start points of J and those of I, in accordance with the mapping $cor_{I\to J}$ found in the trigger. The intervals on the edges are those found in the specification for the corresponding constraint.

Although solving STPs can be done effectively, the overall problem of solving RE-CSPs, which involves the former as a sub-problem, is demonstratively NP-hard, except in the simplest of problems. In particular, the general problem of solving RE-CSPs with relational constraints is NP-hard, as demonstrated in [15].

3.2 Solving RE-CSPs via Clustered Temporal Networks

Concretizations using STPs are "flat" in the sense of eliminating the distinction between intervals that are part of the same repeating event and those that are not. With such concretizations the operations defined on profiles to determine admissibility are "compiled away" into propagation operations on TCSP networks. This section introduces an alternative representation that generalizes the notion of profile and manipulates them explicitly.

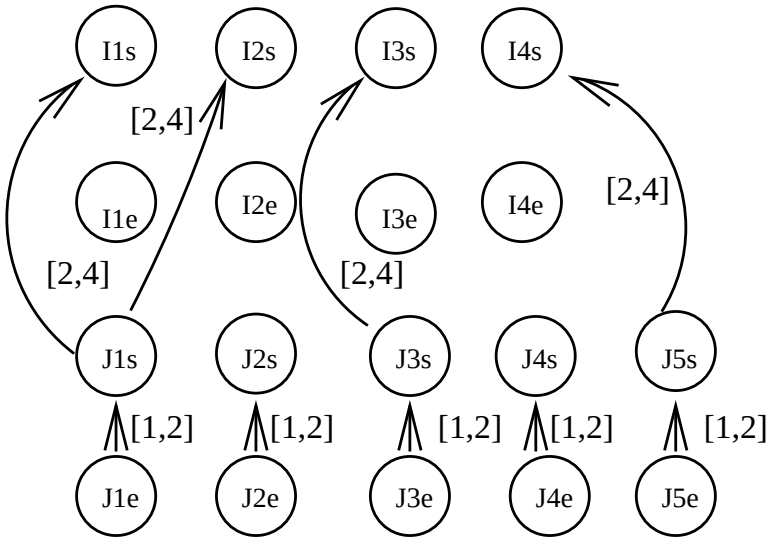

Figure 5: Concretization of a specification into a STN.

This alternative representation is based on the notion of a *partial profile*. A partial profile is a profile whose cells contain interval values. They will be viewed as labels of edges that connect nodes of a network called a Clustered Temporal Network (CTN). Each node in a CTN represents a single repeating event. Triggers define the dimension of each partial profile, and constrain a subset of profile values, based on the correlations established in the triggers. Inverse and composition can be defined on partial profiles, generalizing these operations as defined above for complete profiles.

To illustrate a CTN, consider the following specification:

> I is a non-overlapping repeating event with three occurrences. J has one occurrence. All of the durations of the Is and J is one time unit, and there is a one time unit gap between successive occurrences of the Is. The start of J is one time unit after the start of the first I, one time unit before the start of the second I, and one time unit after the start of the third I.

This specification derives from a class of RE-CSP in which there are, in addition to the variables introduced earlier, other variables which stand for specific profile elements; e.g., $D_{s,s}^{I,J}[1,2]$ stands for the distance $s(J_2) - s(I_1)$. This specification is inconsistent. To detect its inconsistency in a STN concretization, a Single Pairs Shortest Path algorithm is applied, wherein the inconsistency is

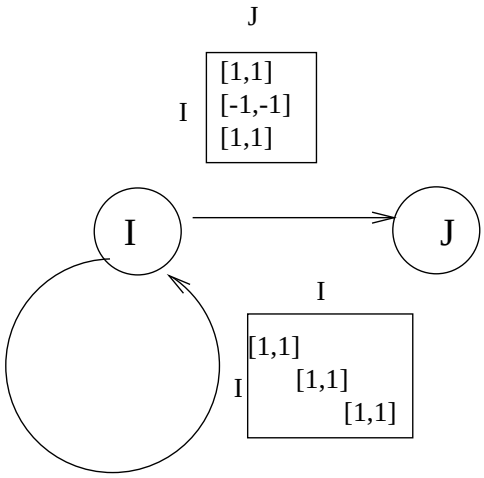

Figure 6: Concretization of inconsistent specification into a CTN.

determined in $O(n^3)$ steps. By contrast, Figure 6 displays the concretization of the specification into a CTN. There are two nodes in the figure, representing the 2 repeating events. Two edges are labeled by partial profiles, a 3×3 matrix representing duration constraints for occurrences of I (the diagonal), and the other a 3×1 matrix representing the binary constraint between the Is and J. Missing profile values are assumed to have the value $[-\infty, \infty]$, signifying no distance constraint between them.

Detecting inconsistency in CTNs can be performed by exploiting the admissibility requirements for profiles discussed earlier. In this example, clearly the monotonicity requirement is violated for the partial profile on the edge between I and J. For partial profiles, this requirement can be roughly stated as follows: there must exist a complete profile (i.e., ones with atomic values) selected from the intervals in the partial profile, which satisfies the requirement. Since the only solution for the profile in question is one in which the values decrease, then increase, the monotonicity requirement is violated. To check for violation of the monotonicity and tri-regional admissibility requirements, $O(N^2M^2)$ checks are made, where N is the number of nodes of the CTN, and M is the largest number of occurrences of any repeating event. This compares with $O(n^3)$ checks on a STP, where n is the number of *start or end points* of all the occurrences of any repeating event. Comparing these worst-case estimates suggests that the ability of CTNs to outperform STPs in practice depends on the ability to "cluster" the reasoning problem into one involving a small number of repeating events. In this case, N, the size of the CTN, will be small, and some efficiency in determining

consistency is expected.

Checking for violations of the monotonicity or tri-regional requirements for admissibility is analogous to performing arc consistency in constraint networks, insofar as only paths of length one are examined. A CTN that adheres to monotonicity requirements is not necessarily a network containing only admissible profiles; the operations of composition and converse must also be preserved. To make a single computation of $P_1 \circ P_2$, where P_1, P_2 are partial profiles, a total of $O(M^3)$ comparisons must be made; thus, an entire CTN is examined in $O(N^2M^3)$ time. Again, if the problem exhibits sufficient clustering, in practice this operation might be performed efficiently.

In this section, we have shown how it is possible to extend existing constraint-based temporal reasoning frameworks for planning to incorporate general temporal constraints concerning the number of time an event is to occur, as well as constraints on the sorts of temporal patterns that can be exhibited by the events. This framework will address the deficiencies faced by existing systems in solving the sorts of problems that were described at the outset. The focus to this point has been on reasoning for the purpose of generating profiles that collectively satisfy a collection of summaries. This semantic relationship between summaries and profiles has a converse relationship which forms the basis for reasoning from a profile or set of profiles, to a summary. This relationship is examined in the next section.

4 Mining Temporal Information

Thus far, we have focused on the use of general temporal knowledge to generate temporal plans from requests formalized as RE-CSP specifications. The focus in this section turns to the problem of extracting useful information from the results of executing plans. The simple temporal ontology introduced above recognizes events as the sole entity associated with time, such time being expressed as durations or ordering relationships. A temporal domain model is a set of propositions expressing duration and ordering constraints over a set of events. The interest in this section is using information about executions to refine a temporal domain model, or to infer new information.

A number of reasoning tasks fit into the research area referred to as *mining temporal data*, including *event detection* [10] (inferring the time certain important events occur), *trend discovery* [6] (mining significant changes in the value of some parameter) and *activity monitoring* [8] (noticing when a change in the behavior of something has occurred). The interest in this paper has been on pure temporal reasoning, which in the area of data mining will include mining duration and temporal ordering information. In the KDD literature, this is often referred to as *mining interesting episodes in temporal data* [12].

The problem to be considered here requires a system to take an execution log consisting of an instantiation of a executed plan (i.e. a set of pairings of

times to events), and generate useful summaries of the temporal information contained in it, or verify that some proposition, expressed as general temporal knowledge by a human user, is true. This summary will be expressed in the same language as that of the temporal domain model; hence summaries can be used to update the model itself. For example, a summary might detect a precedence ordering between events that was previously not expressed in the domain model. Similarly, observed distances between pairs of events can be expressed as a summary and added as new knowledge to the domain model. Consequently, the system is continuously improving the model of time used to generate plans as the result of its own planning activities.

4.1 Mining as operations on profiles

Instantiations are *flat* representations of temporal data, insofar as they have a representation as a table of rows and columns. Profiles provide an intermediate level of structure to these data by providing concise representations of distance and ordering information. Two kinds of grouping occur in the transition from instantiations to profiles: events are paired off based on their type, and also paired off based on end points. This transformation allows for patterns to be revealed. Summaries, which are the results of further operations on profiles, are concise expressions of the patterns revealed in profiles.

Recall the operations of inversion and composition of profiles introduced earlier. These were used in the definition of admissible profiles, which formalized the solution to the planning problem involving general temporal constraints. To this set of operations we introduce others that produce summaries out of profiles.

First, we distinguish between two kinds of summaries: *value* and *correlation*. Let P_O be the space of admissible order profiles, and P_D be the space of admissible distance profiles. One kind of simple value summary can be viewed as a function $V_D : P_D \rightarrow P(\mathcal{I})$, i.e. from the space of possible distance profiles to a subset of the Integers. Informally, these summaries return a subset of the values in the distance profile. By contrast, qualitative value summaries are functions of the form $V_O : P_O \rightarrow P(\mathcal{A})$, where $\mathcal{A}$ is the set of Allen relations. Thus, these summaries return the set of Allen relations in a profile. It is assumed that there are functions for further modifying the sets returned by simple value summary. For example, let A be a set of the Integers, and let $min(A), max(A)$ be the minimum and maximum of the values in A. Let the function $range(A)$ return the interval $[min(A), max(A)]$, i.e., the range of values within A. Similarly, given a set A of Allen relations in a profile, let $overlaps - some(A)$ return the set of Allen relations in A that are not b or bi. These auxiliary functions provide additional means of summarizing the values in the set returned by a value summary.

Correlation summaries, by contrast, will be viewed as operations that return *sub-profiles* of a profile. If a profile is viewed as a set of triples $\langle I_i, J_j, val \rangle$, a sub-profile of P is any subset of this set of tuples. Hence, sub-profiles are

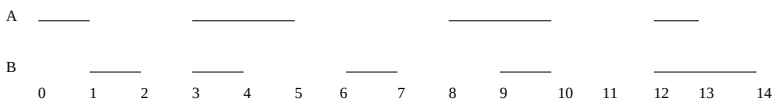

Figure 7: An instantiation of two repeating events

m	b	b	b	b
bi	si	b	b	b
bi	bi	bi	fi	b
bi	bi	bi	bi	s

1	3	6	9	12
-2	0	3	6	9
-7	-5	-2	1	4
-11	-9	-6	-3	0

Figure 8: Order and distance ($s(B) - s(A)$) profiles for example in Figure 7

profiles, and correlation summaries are functions $C : P \rightarrow P$, i.e., from profiles to profiles.

Here are some examples of each kind of summary. The instantiation in Figure 7 will be used to illustrate. Figure 8 contains the order and one distance profile generated from the instantiation. One example of a simple qualitative value summary would be **the set of all relationships between A and B**. This is a function that, given an order profile P_O for A and B, returns the set of Allen relations in P_O; in the example, this set would be $\{b, bi, m, si, fi, s\}$. Similarly, the summary **the distance between the start of some occurrence of A and the start of some occurrence of B** is a value summary that returns $\{-11, -9, -7, -6, -5, -3, -2, 0, 1, 3, 4, 6, 9, 12\}$, i.e., all the values in the distance profile. An example of a simple correlation summary would be: **all the ordering relations between the first and second occurrences of A and B**. This summary would return a 2×2 sub-profile consisting of the upper left part of the order profile in the figure.

Most summaries of interest can be viewed as a series of compositions of value and correlation summaries. For example, consider the summary: **the range of distances between the start of every occurrence of A and the start of the occurrence of B in closest proximity**. Intuitively, this is the result of examining each row of the distance profile in Figure 8, extracting the value(s) closest to 0, expressing the result as an range (interval) of values; in the example, the result would be $[0, 1]$. Similarly, **the set of pairs of occurrences of As and Bs in closest proximity, and the distances between their start times**, can be viewed as a correlation summary returning the sub-profile comprised of elements along a diagonal of the input profile; specifically, the set of triples $\{\langle A_1, B_1, 1\rangle, \langle A_2, B_2, 0\rangle, \langle A_3, B_4, 1\rangle, \langle A_4, B_5, 0\rangle$. Notice that this summary reveals the pattern of alternating distances of 1 and 0 between start

times of the correlated occurrences, potentially of interest to the viewer of the data.

4.2 Mining interesting temporal episodes

A simple two-step procedure for mining interesting episodes from an instantiation is, first, to generate a profile from the instantiation, and second, produce (or verify the truth of) a summary from information found in the profile. This basic procedure can be refined based on the degree to which the user guides the mining process. Later in this section, we sample from a range of degrees of user guidance.

First, notice that generating a single profile from an instantiation takes $O(n^2)$ time, where n is the number of occurrences of the event with the most occurrences. More specifically, if E has n occurrence, and F has m occurrences, it clearly takes $n \times m$ difference calculations to generate a distance profile. For order profiles, each entry is generated by comparing both end points of one interval with both end points of another. Hence, the overall cost of populating an order profile is $4(m \times n)$. This cost can be reduced if E and F are both sequences of non-overlapping occurrences; then, it is possible to apply admissibility criteria for profiles to generate values directly. For example, truths such as *if I_n is before J_m, then it is before every occurrence of J after J_m* can be applied to determine some of the entries without explicitly comparing end points. Thus, in the first row of the order table in Figure 8, once the first b is detected, the remaining values in the row must be b in order for the profile to be admissible. In effect, the comparisons are thereby limited to entries that will be along the diagonal of the profile.

Once the profile has been built for the indicated events, the remainder of the calculation involves investigating the temporal patterns exhibited by them. Let us consider three degrees of user guidance in the process: complete, partial, and none. Complete guidance occurs when a user wishes to know whether a profile satisfies a summary. Recall that a distance summary consists of the following parts:

1. A pair of events A, B;
2. A mapping expressed as a pair of quantifiers, one each for occurrences of A or B;
3. A pair of end points $x, y \in \{s, e\}$; and
4. An interval $[l, b]$ of values

An order summary substitutes a set of Allen relations for items (3) and (4) in the list. Therefore, a completely guided mining activity would require the user to supply each of the four (3) components of a distance (order) summary. Notice that the entire process of verifying the truth of a distance or order summary can

be conducted efficiently for many of the examples we've been presenting. For example, the interest might be in determining how often the start of some B immediately (i.e. within 1 time unit) follows the end of some A. This involves examining each row of the distance profile in Figure 8, and counting how many times the value 1 appears. In general, two complete perusal of a matrix suffices for verification, one for creating the profile, the other for verifying the summary. (A more efficient procedure would interleave the creation and verification process.)

A simple way of viewing partial user guidance is when a subset of the 4 components of a distance summary (or the 3 of an order summary) are left unspecified. Once this happens, issues related to criteria for the automatic detection of *interesting* temporal information arise. What makes temporal (duration or order) information "interesting"? We distinguish between two criteria.

First, there is a tendency for occurrences in close proximity to be the focus of mining episodes. For example, the gap between *consecutive* occurrences of the same event tend to be more interesting than the gap between occurrences separated by other occurrences. Similarly, occurrences that happen around occurrences of other events tend to hold more interest than pairs of occurrences separated by longer distances. The reason is usually due to the fact that mining temporal summaries is often related to the goal of discovering causal relationships, and these relationships tend to be revealed among occurrences in close proximity. For example, the occurrence of high fever following the administration of a certain treatment would be interesting information for a physician (from a causal standpoint) only if the events occur in close proximity; if the two events are separated by more than, say a few days, a different causal relationship (involving other events) would be suggested.

Second, there is interest in temporal relationships only if there is a duration or order information that falls into a pattern. A number of patterns are possible, involving *frequency*, *periodicity*, or *simple repetition*. If fever follows a therapy treatment with low frequency, there is less interest than if it is followed with high frequency. Similarly, if there is no overlap between certain events, say, between occurrences of maintenance tasks for different power generation units, then this might fit the criteria for being interesting. Sometimes, it is the range of values that is interesting; for example, if the set of all the durations between two events is in $[0, 1]$ a causal relationship between the occurrences might be suggested.

What are the requirements for a system that can find interesting temporal patterns without guidance of any kind? In addition to a temporal model of proximity and frequency, a completely automated temporal knowledge discovery system will need a model of "interesting pairings" of temporal entities. This amounts to having a robust model of what temporal entities are naturally paired with others. For example, in the medical domain, what makes *administering drug* events and *fever* states in patients conducive to associations is the fact that there may be causal relations between events similar to the drug administering

event and the onset of fever. Notice that there are $O(\binom{M}{2})$ profiles associated with an instantiation of M events. Hence, the task of generating interesting pairings from raw data grows exponentially with the number of event types. The work of Shahar [21] addresses the problem of building an ontology of temporal entities which guides the automated tool in the search of interesting pairings.

Finally, the preceding examples have been restricted to cases which require producing information about binary relationships between events. This treatment can be extended to patterns involving more than 2 events. For example, suppose the interest is in determining the frequency of the pattern $s(I_i) - s(J_j) \in [1,1] \wedge s(J_j) - s(K_k) \in [1,1]$. This pattern is that of the start of an I immediately following the start of a J which, in turn, immediately follows the start of some K. Clearly, this query can be solved by creating profiles for both $s(I_i) - s(J_j)$ and $s(J_j) - s(K_k)$, and examining the values therein. In general, in many cases, it is possible to view the task of finding interesting episodes among a set of $n > 2$ events as a set of examinations of (binary) profiles.

This section has described informally the process of extracting useful summaries of temporal information in the form of instantiations of events. The summaries arise from a two step procedure of, first, extracting the relevant profile(s), and second, extracting general temporal knowledge from them. Since the knowledge extracted is of the same logical form as the knowledge used to generate temporal plans, the extracted knowledge can be used to update the knowledge found in the temporal domain model used by the planner. Consequently, we have finished our description of a closed-loop system for temporal planning and plan execution analysis.

5 Conclusions

This paper has proposed a single representation of temporal information for supporting a more robust framework for formulating planning problems, as well as to formulate summaries of execution traces of plans from execution logs stored as temporal databases. This framework has the potential for proving useful within systems that combine planning and plan execution, which are becoming more common as AI technology continues to mature. The approach to realizing this framework builds upon existing frameworks based on the CSP representation of temporal reasoning problems. The proposed framework is based on a simple, intuitive distinction between temporal specifications, summaries, profiles and instantiations. The operations that transform specifications into instantiations to solve planning problems can be inverted to formulate and solve the problem of mining interesting temporal information.

The framework proposed in this paper uses the classical constraint-based representation of temporal knowledge. The classical framework considers all solutions to be of equal value with respect to satisfying the requirements for solving the problem, whereas the information extracted from the trace data

from executions might suggest a ordering of solutions based on frequency of occurrence. For example, although the temporal domain model might assert that a certain pair of events can happen between 5 and 10 time units apart, based on observations it can be determined that the distance is usually either 9 or 10 units apart. This additional knowledge could be used in the plan generation phase to prefer solutions that reflect past observations such as this. There are generalizations of of the classical CSP framework, e.g. the Semiring CSP representation [2], that allow for the generation of solutions to CSPs that adhere to local preference criteria. We are currently investigating such extensions of the proposed framework.

6 Acknowledgments

The authors thank Yuval Shahar, Shubha Chakravarty, and Paul Morris for stimulating discussions of themes discussed in this paper.

References

[1] J. Allen. Maintaining knowledge about temporal intervals. In *Readings in Knowledge Representation*, pages 510–521, Morgan Kaufman Publishers, Inc., 1983.

[2] S. Bistarelli, U. Montanari, and F. Rossi Semiring-based Constraint Solving and Optimization. *Journal of the ACM*, 44(2):201-236, March, 1997.

[3] J. Bresina. Telescope loading: A problem reduction approach. In *Proceedings of the Third International Symposium on Artificial Intelligence, Robotics, and Automation for Space*, Pasadena, CA, Jet Propulsion Lab, 1994.

[4] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1-3):61–95, 1991.

[5] M. desJardins, E. Durfee, C. Ortiz Jr, and M. Wolverton. A survey of research in distributed, continual planning. *AI Magazine*, 20(4):13–22, Winter 1999.

[6] G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. In *Proceedings of KDD-99*, pages 43–52, San Diego, CA, USA, 1999.

[7] H. Eriksson, Y. Shahar, S. W. Tu, A. R. Puerta, and M. A. Musen. Task modeling with reusable problem-solving methods. *Artificial Intelligence*, 79(2):293–326, 1995.

[8] T. Fawcett and F. Provost. Activity monitoring: Noticing interesting changes in behavior. In *Proceedings of KDD-99*, pages 53–62, San Diego, CA, USA, 1999.

[9] D. Frost and R. Dechter. Maintenance scheduling problems as benchmarks for constraint algorithms. Manuscript, 2000.

[10] V. Guralnik and J. Srivastava. Event detection from time series data. In *Proceedings of KDD-99*, pages 33–42, San Diego, CA, USA, 1999.

[11] F. Kabanza, J-M. Stevenne, and P. Wolper. Handling infinite temporal data. *Journal of Computer and System Science*, 51:3–17, 1995.

[12] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovering frequent episodes in sequences. In *Proceedings of the First Int'l Conference on Knowledge Discovery and Data Mining*, pages 210–215, Montreal, Quebec, Canada, 1995.

[13] R. Morris and L. Khatib. An interval-based temporal relational calculus for events with gaps. *Journal of Experimental and Theoretical Artificial Intelligence*, 3:87–107, 1991.

[14] R. Morris and L. Khatib. Constraint reasoning about repeating events: Satisfaction and optimization. *Computational Intelligence*, 16(5), July 2000.

[15] R. Morris and P. Morris. On the complexity of reasoning about repeating events. In *Proceedings of the 7th International Conference of Knowledge Representation and Reasoning*, 2000.

[16] R. Morris, W. Shoaff and L. Khatib. Path consistency in a network of non-convex intervals. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 124–134, 1995.

[17] R. Morris, W. Shoaff and L. Khatib. Domain independent temporal reasoning about repeating events. *Computational Intelligence*, 16(2), 1996.

[18] N. Muscettola, P. Nayak, B. Pell, and B. Williams. Remote agent: to boldly go where no ai system has gone before. *Artificial Intelligence*, 103:5–48, 1998.

[19] B. Padmanabham and A. Tuzhilin. Pattern discovery in temporal databases: A temporal logic approach. In *Proceedings of the Second Int'l Conference on Knowledge Discovery and Data Mining*, pages 351–354, 1996.

[20] Y. Shahar. A framework for knowledge-based temporal abstraction. *Artificial Intelligence*, 90(1-2):79–133, 1997.

[21] Y. Shahar and C. Cheng. Model-based visualization of temporal abstractions. *Computational Intelligence*, 16(5), 2000.

[22] Y. Shahar and M. A. Musen. Knowledge-based temporal abstraction in clinical domains. *Artificial Intelligence in Medicine*, 8(3):267–298, 1996.

[23] Y. Shoham. Temporal logics in ai: Semantical and ontological considerations. *Artificial Intelligence*, 33(1):89–104, 1987.

[24] P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56:72–99, 1983.